НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра автоматики та управління в технічних системах

До захисту допущено:

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____20__ р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Комп'ютеризовані системи управління»
спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології»
на тему: «Автоматизована система надання рекомендацій з підбору комплектуючих персонального комп'ютера»

Виконав (-ла):
студент (-ка) IV курсу, групи ІА-61
Шадлер Микола Андрійович _____

Керівник:
Асистент
Дорога-Іванюк Олена Олександрівна _____

Рецензент:
доцент кафедри ПЗКС ФПМ, к.т.н.
Цуркан Василь Васильович _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2020 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра автоматики та управління в технічних системах

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 151 «Автоматизація та комп'ютерно-інтегровані технології»

Освітньо-професійна програма «Комп'ютеризовані системи управління»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____20__ р.

ЗАВДАННЯ

на дипломний проєкт студенту

Шадлеру Миколі Андрійовичу

1. Тема проєкту «Автоматизована система надання рекомендацій з підбору комплектуючих персонального комп’ютера», керівник проєкту Дорога-Іванюк Олена Олександрівна, асистент, затверджені наказом по університету від «07» травня 2020 р. №1081-с

2. Термін подання студентом проєкту _____09_червня_2020_року____

3. Вихідні дані до проєкту

Мова програмування C#, бібліотеки EntityFramework, MvvmLightLabs, середовище розробки Visual Studio, обраний фреймворк .NET Core, СУБД – MSSQL, графічна підсистема WPF, система керування версіями.

4. Зміст пояснювальної записки

Опис предметної області, аналіз існуючих рішень, опис розроблюваної рекомендаційної системи, використані шаблони та підходи проектування, аналіз розробленої системи, тестування розробленого рішення 8. Впровадження та використання розробленого додатку

5. Перелік графічного матеріалу

Діаграма використання, діаграма компонентів, діаграма структури бази даних, діаграма класів

6. Дата видачі завдання _____02_лютого_2020_року_____

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вибір тематичного напрямку та узгодження теми дипломного проєкту	27.02.2020	
2	Аналіз предметної області	13.04.2020	
3	Аналіз існуючих рішень	20.04.2020	
4	Аналіз вимог до програмного забезпечення	27.04.2020	
5	Вибір технологій розробки	03.05.2020	
6	Розробка автоматизованої системи	10.05.2020	
7	Налагодження та перевірка програми	17.05.2020	
8	Оформлення пояснювальної записки	20.06.2020	
9	Передзахист дипломного проєкту	25.05.2020	
10	Доопрацювання пояснювальної записки та підготовка презентації	30.05.2020	
11	Захист дипломного проєкту	09.06.2020	

Студент Микола ШАДЛЕР

Керівник Олена ДОРОГА-ІВАНЮК

АНОТАЦІЯ

Шадлер М.А. Автоматизована система надання рекомендацій з підбору комплектуючих персонального комп’ютера. КПІ ім. Ігоря Сікорського, Київ, 2020.

Проєкт містить 60 с. тексту, 9 рисунків, 19 таблиць, посилання на 17 літературних джерел, додатки та 4 конструкторських документів.

Ключові слова: C#, EntityFramework, VisualStudio, система керування версіями, персональний комп’ютер, комплектуючі.

Об’єктом розробки є автоматизована система надання рекомендацій з підбору комп’ютерних комплектуючих.

Мета розробки – спрощення процесу проектування персонального комп’ютера для недосвідчених користувачів.

У дипломному проєкті проведено ретельний аналіз та вибір технологій розробки системи, які надають можливість швидкої та якісної розробки програмного забезпечення. Розроблено автоматизовану систему надання рекомендацій з підбору комплектуючих персонального комп’ютера з використанням фреймворку .NET Core, мови програмування C# та графічної підсистеми WPF.

Отримані результати можуть бути корисними при створенні та покращенні аналогічних чи подібних систем.

SUMMARY

Shadler M.A. Automated system for providing quidance on the selection of personal computer components. Igor Sikorsky KPI, Kyiv, 2020.

The project contains 60 pages, 9 pictures, 19 tables, references to 17 literature sources, appendices and 4 design documents.

Keywords: C #, EntityFramework, VisualStudio, version control system, personal computer, components.

The object of development is an automated system for providing recommendations for the selection of computer components.

The purpose of the development is to simplify the process of designing a personal computer for inexperienced users.

The graduation project has a thorough analysis and selection of system development technologies that provide the ability to quickly and efficiently develop software. An automated system for providing quidance on the selection of personal computer components has been developed using the .NET Core framework, the C # programming language and the WPF graphics subsystem.

The obtained results can be useful in creating and improving analogous or similar systems.

Пояснювальна записка до дипломного проєкту на тему: «Автоматизована система надання рекомендацій з підбору комплектуючих персонального комп'ютера»

Київ – 2020 року

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ЦП – центральний процесор

ПК – персональний комп’ютер

ГП – графічний процесор

ПЗ – програмне забезпечення

ООП – об’єктно орієнтоване програмування

БД – база даних

СУБД – система управління базою даних

ККД – коефіцієнт корисної дії

MVC – Model View Control

CPU – Central processing unit

RAM – Random Access Memory

SIMM – Single In-line Memory Module

DIMM – Dual In-line Memory Module

SO DIMM – Small Outline DIMM

DDR – Double Data Rate

AMD – Advanced Micro Devices

VCS – Version Control System

SVN – Subversion

VS – Visual Studio

WPF – Windows Presentation Foundation

UWP – Universal Windows Platform

UC – User Control

MVVM – Model View ViewModel

HDMI – High Definition Multimedia Interface

DVI – Digital Visual Interface

UPS – Uninterruptible Power Supply

HDD – Hard Drive Disc

SSD – Solid State Drive

SSHD – Solid State Hard Drive

TDD – Test Driven Development

SLI – Scalable Link Interface

ORM – Object-Relational Mapping

BIOS – Binary Input-Output System

BGA – Ball Grid Array

LINQ – Language-Integrated Query

MSSMS – Microsoft Server Management Studio

EF – Entity Framework

PCIe – Peripheral Component Interconnect Express

GDDR – Graphics Double Data Rate

USB – Universal Serial Bus

RAID – Redundant Array of Independent Disks

ВСТУП

Персональний комп’ютер – настільна електронна обчислювальна машина, що призначена для обробки та зберігання інформації. Використовується на різноманітних підприємствах як основний чи прикладний інструмент праці або у домогосподарствах для забезпечення дозвілля. Обирається таким чином, щоб його конфігурація та ціна відповідала тим задачам, які будуть поставлені перед користувачем.

Наразі у вільному продажу існує безліч готових рішень, що задовольняють потреби середньостатистичної людини, але не завжди мають збалансовані характеристики. Частіше всього цим користуються магазини роздрібної торгівлі для підвищення рівня продажу комплектуючих та отримання додаткового прибутку, включаючи до ціни додаткову вартість за зборку. Крім того, готові рішення не враховують подальші зміни, що може внести користувач задля підвищення продуктивності або розширення функціональних можливостей комп’ютера або взагалі забороняють такі дії під страхом втрати гарантії. Рішенням цих проблем є самостійне проектування та збирання персонального комп’ютера з використанням різних інформаційних ресурсів, що мають у собі інструкції для користувача, щодо правильного підбору та монтажу компонентів. У зв’язку з чим, будуть набирати популярність додатки, що мають значно спростити процес підбору комп’ютерних комплектуючих, а отже, є актуальним розробка даного програмного рішення.

Метою даної роботи є створення автоматизованої системи надання рекомендацій з підбору комплектуючих персонального комп’ютера. Це – програмний комплекс, що містить у собі підказки, щодо призначення кожного компонента, сумісності та збалансованості з іншими, що забезпечує найкраще співвідношення ціни-продуктивності та дозволяє врахувати подальші зміни у конфігурації.

Для досягнення поставленої мети були вирішені наступні завдання:

— огляд існуючих рішень, що полегшують процес проектування персонального комп’ютера;

— формування вимог до розроблюваного програмного забезпечення та вибір засобів розробки;

— розробка рекомендаційної та фільтраційної систем, що сприятимуть більш швидкому вирішенню поставленої задачі з проектування персонального комп’ютера;

— аналіз та тестування розробленого ПЗ.

Для створення даного проекту було використано мову програмування C# та супутні технології, такі як середовище розробки VisualStudio та платформа .NET Core, а також графічна підсистема, що надається ними, WPF.

1 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Проблема вибору

Так як ПК є системою, дуже важливо збирати його з огляду на максимально можливу до врахування кількість параметрів кожного з компонентів. Людині необізнаній у будові комп’ютера важко осягнути такий об’єм інформації, тож стає проблематично сконфігурувати ПК самотужки. Значно полегшує дану задачу комплексний підхід, а саме поступове вивчення області відповідальності кожного компонента з записом його характеристик до блокноту чи то іншого носія інформації.

Враховуючи сучасні тенденції до діджиталізації є нормальним спроби людини знайти додаток, що спростить цю роботу, та скриє у собі більшість характеристик шляхом автоматичної фільтрації. Тобто, людина зможе сконцентруватися на виборі комплектуючих базуючись на її користувацькому знанні ринку.

1.2 Складові об’єкту дослідження

Типовий персональний комп’ютер складається з наступних компонентів:

— процесор;

— оперативна пам’ять;

— материнська плата;

— блок живлення;

— постійна пам’ять;

— відеокарта;

— корпус.

Кожен з них має певний набір характеристик, що мають бути узгоджені між собою для того, щоб мати змогу працювати одне з одним.

1.2.1 Центральний процесор

Центральний процесор – це високоінтегрована надвелика інтегральна схема складної структури в єдиному напівпровідниковому кристалі. В англомовній літературі ЦП називають CPU – Central Processor Unit або main processor. Здійснює координацію потоків даних і їх обробку. Апаратура ЦП забезпечує ефективний і гнучкий захист пам'яті, контрольований доступ до ресурсів оперативної системи, ізоляцію індивідуальних прикладних програм, малий час реакцій на переривання. Для розміщення процесора на материнській платі використовується спеціальне гніздо, зване Socket.

Проблема теплообміну стала актуальною з підвищенням робочої тактової частоти процесорів і жорсткістю технологічних норм при виробництві кристалів. Для охолодження процесора використовується малогабаритний вентилятор, встановлений на радіатор ЦП – процесорний кулер. Ця система значно знижує температуру процесора що дозволяє йому працювати не вдаючись до троттлінгу (скидання частоти заради зниження температури).

Найчастіше, саме потужність процесора визначає потужність системи. Головними характеристиками ЦП є:

- тактова частота, енергоспоживання;
- норми літографічного процесу (технічний процес, техпроцес);
- архітектура;
- кількість ядер;
- об’єм кеш-пам’яті;
- сокет;
- к-сть ядер;
- кількість потоків.

Залежно від потужності процесора обирається відповідна йому за сокетом материнська плата та за частотою і поколінням оперативна пам’ять. Енергоспоживання враховується при виборі блока живлення.

1.2.2 Оперативна пам’ять

ОЗП (RAM – Random Access Memory «Пам'ять довільного доступу», тобто в будь-який момент часу доступ може здійснюватися до довільно вибраної комірки) – робоча пам'ять, забезпечує можливість оперативного обміну інформацією, в тому числі в процесі виконання операцій. Призначена для зберігання змінної інформації, допускає зміну свого вмісту в ході виконання процесором обчислювальних операцій. Вона забезпечує режими запису, зчитування і зберігання інформації. Це – енергозалежна пам'ять, інформація після виключення ПК з ОЗУ стирається, а швидкість доступу до неї значно вища ніж до постійної(енергозалежної).

За типом розміщення модулів пам’яті бувають:

– SIMM (Single In-line Memory Module), односторонні, ставляться парами, напруга живлення 5 В.

– DIMM (Dual In-line Memory Module), двусторонні, можно ставити по одинці, напруга живлення 5 та 3,3 В.

Як підвид DIMM можна виділити стандарт SO DIMM (Small Outline DIMM), що має менших розмір та розраховано на його використання у компактних ПК або ноутбуках.

Найважливішим для користувача параметрами є:

– покоління оперативної пам’яті;

– частота;

– схема латентності (таймінги);

– форм-фактор (інтерфейс підключення).

Перші три параметри напряму впливають на швидкість роботи пам’яті, а останній визначає фізичний інтерфейс підключення (роз’єм) до материнської плати та її розмір, що накладає додатковий фільтр на її вибір.

1.2.3 Материнська плата

Материнська плата – основний елемент персонального комп'ютера від якого залежить функціональність комп'ютера. Будучи багатошаровою пластиною з провідними доріжками забезпечує зв'язок між іншими компонентами, а саме процесором, оперативною та основною пам'яттю, відеокартою, блоком живлення та периферією. Взаємодія даних пристроїв забезпечується чипсетом, що складається з двох частин – північного (northbridge) та південного (southbridge) мостів. Зазвичай вони розташовані на різних мікросхемах. Є основними елементами, що визначають особливості материнської плати і перелік можливих до підключення пристроїв. Також плата містить перелік роз'ємів для підключення ЦП, графіного адаптера, оперативної та постійної пам'яті, плат розширення. Більшість має вже вбудовану аудіокарту та мережеву плату, підтримують бездротові технології на кшталт Bluetooth та 802.11 (Wi-Fi).

На материнській платі розміщуються такі елементи:

– сокет або гніздо ЦП. За допомогою контактних ніжок або пластинок процесор з'єднується з сокетом. В основному, передбачається можливість швидкої заміни, але у компактних ПК може бути припаяним до плати (BGA);

– мікросхема BIOS, що містить у собі внутрішні інструкції для забезпечення первинної роботи. Дана система надає користувачу можливість за допомогою графічного інтерфейсу змінювати конфігурацію обладнання: зміну частоти процесора або оперативної пам'яті, керування схемами латентності або живленням. Зміна даних параметрів може призвести до виходу з ладу певних пристроїв, тож вносити зміни рекомендується тільки досвідченим користувачам. Поруч розташовується батарейка, що підтримує роботу годинника задля синхронізації часу у вимкненому стані;

– слоти модулів оперативної пам'яті. Потрібно враховувати їх кількість, форм-фактор та підтримуване покоління пам'яті. Наразі найпопулярнішими є DDR3 та DDR4, існують варіанти, що підтримують обидва одразу, але тоді вони працюють окремо. В основному, материнські плати мають два або чотири слоти

одного покоління та підтримують роботу з пам’яттю у одно- або двопоточному режимах;

– PCIe – роз’єми для встановлення додаткових пристроїв накшталт відеокарти, карт розширення та, іноді, постійної пам’яті;

– SATA та M.2 – інтерфейси для підключення накопичувачів, як SSD так і HDD. Зазвичай їх декілька і крім підключення дисків надають змогу встановлювати дисководи;

– роз’єми живлення існують у двох основних типах – ATX 24-контактний і 4-контактний з додатковою лінією 12V. Часто містить модуль для регулювання напруги, що складається з конденсаторів та силових транзисторів і займається тим, що стабілізує і фільтрує напругу з блоку живлення;

– задня панель, що дозволяє підключать периферійні пристрої – монітори, клавіатури і миші, мережеві, аудіо та інші USB пристрої.

Окрім вищеперерахованих роз’ємів є піни для підключення кулерів, системного динаміка

Має наступні характеристики, що важливо враховувати при зборі комп’ютера:

– форм-фактор;

– чипсет (chipset);

– сокет (socket);

– підтримка певного покоління оперативної пам’яті;

– наявність загальновживаних інтерфейсів підключення периферійних пристроїв та плат розширення.

Підсумовуючи, по-перше, сокет материнської плати має відповідати сокету процесора, бо інакше його встановлення у процесорний роз’єм неможливе фізично, через різницю інтерфейсів (ніжки та контактні площадки). По-друге, підтримуване покоління оперативної пам’яті – відмінності у фізичному роз’ємі та можливості процесора працювати з ним, об’ємі бажаної пам’яті та можливій кількості потоків. Чипсет відповідає за максимальну потужність персонального комп’ютера, а форм-фактор впливає на вибір корпусу.

1.2.4 Постійна пам’ять

На даний момент є два різновиди енергонезалежної пам’яті, що використовується у ПК, це HDD та SSD.

HDD – Hard Disc Drive – жорсткий диск, що побудований з використанням магнітів. Інформація зберігається на шарі феромагнітного матеріалу, нанесеного на тверді, у формі диску, пластини. Запис/зчитування відбувається за допомогою магнітної головки, що ковзає у безпосередній близькості до цих пластин, які, в свою чергу, приводяться у дію моторчиком та розкручуються до 5400 або 7200 оборотів за хвилину. Жорсткі диски є відмово стійкими за умови їх правильного зберігання та мають досить невелику ціну за гігабайт. Недоліком є можливість втрати працездатності при ударі корпусу. Втрачені таким чином дані можуть бути частково відновлені у спеціально обладнаних лабораторіях.

SSD – Solid State Drive – запам’ятовуючий пристрій, що базується на основі мікросхем та контролера та не містить, на відміну від HDD, рухомих механічних частин. Найчастіше використовується в компактних пристроях: ноутбуках, нетбуках, смартфонах через їх значно менший розмір. Наприклад, форм-фактор M.2 може мати розміри інтегральної схеми 22 мм х 30 мм.

Також існують так звані SSHD – гібридні жорсткі диски, що мають у собі невеликий обсяг твердотільного накопичувача в ролі кешу.

Переваги перед твердими дисками (HDD):

– відсутність рухомих частин;

– висока швидкість запису/зчитування, що перевершує пропускну здатність інтерфейсу твердого диска;

– повна відсутність шуму (немає рухомих частин і вентиляторів охолодження);

– висока механічна стійкість;

– малі габарити і вага;

З недоліків можна виділити обмежену кількість циклів перезапису – від 10 до 100 тисяч разів у MLC (Multi-level cell) та, відповідно, більш дорогої SLC (Single-level cell). Також, даний тип накопичувачів має певні проблеми

сумісності з застарілими версіями ОС, що не враховують особливості роботи з ними, та штучно зменшують ресурс шляхом виконання стандартних операцій з підтримки працездатності жорстких дисків.

Ціна гігабайту SSD-накопичувачів значно вища за ціну гігабайта HDD-накопичувача. Тож часто можна бачити комп'ютери з одночасно двома накопичувачами – SSD меншого об'єму, для встановлення на неї ОС та певних програм, щоб прискорити швидкість їх роботи, та HDD для зберігання різного роду файлів.

Виділимо основні характеристики:

- об'єм пам'яті;
- кеш;
- інтерфейс підключення
- форм-фактор
- швидкість оборотів шпинделя;
- середня швидкість пошуку.

1.2.5 Графічний адаптер

Графічний адаптер (відеокарта) – пристрій, що перетворює графічне зображення, що зберігається у пам'яті в сигнал, що може бути виведено на екран. Наразі є два популярні варіанти відеокарт – дискретна або інтегрована.

Що стосується сучасних інтегрованих відеокарт, то вони, як правило, вбудовуються у процесор, не мають власної оперативної пам'яті, а тому користуються загальною. Але є обмеження щодо об'єму, тож це не може розцінюватися як перевага. Це значно впливає на швидкість роботи в гіршу сторону даних відеокарт, через збільшення часу обміну даними. Використовуються у бюджетних рішеннях, коли користувач не має потреби у відображенні складних графічних сцен, будь то відео з великим розширенням або сучасні ігри. Мають знижене енергоспоживання, тому знайшли популярності у компактних комп'ютерах та ноутбуках економлять місце за рахунок

відсутності окремого чипу, а тому й тепловідвідної пластини та додаткової системи активного охолодження.

Дискретна відеокарта є найбільш потужною, переважна більшість обмінюється даними через високошвидкісну шину PCIe. Виконується у вигляді текстолітової пластини, на якій розпаяні усі частини. За своєю архітектурою дискретний відеоадаптер нагадує ПК, бо також має PU (processor unit), постійну та оперативну пам'ять. Варто зазначити, що дискретні відеокарти не обов'язково виконуються у вигляді окремих пристроїв. У ноутбуках дискретною графікою зветься окремий від процесора чіп, що припаяний до материнської плати та не є частиною інших компонентів. Як правило, дискретні відеоадаптери мають власну оперативну пам'ять, що також характеризується частотою та об'ємом. Виконані як окремий набір системної логіки, вони є значно продуктивнішими та складнішими. Значну роль у цьому відіграє відсутність необхідності до розподілення доступу до оперативної пам'яті. Відеокарта не займає її своїми даними, не чекає черги на доступ, обмінюється без проміжних інтерфейсів, таких як PCIe. Часто у сучасних рішеннях використовуються такі технології як SLI від Nvidia і CrossFire від AMD, що дозволяють використовувати декілька графічних адаптерів паралельно для вирішення одної задачі.

З основних характеристик, що потрібно враховувати при виборі графічного адаптера можна виділити ГП, ОЗП та відео порти.

Графічний процесор (GPU) – аналог ЦП, що займається формуванням зображення, що виводиться, розраховує та оброблює дані для тривимірної графіки. У той час, як процесор будує примітиви, графічний процесор накладає на них текстури. Саме тому, дуже важливо, щоб ЦП та ГП були одного класу потужності, бо якщо один з них не буде встигати виконувати свою частину функціоналу, інший почне простоювати. Часто, сучасні ГП значно перевершують ЦП як числом транзисторів так і обчислювальною потужністю завдяки наявності універсальних обчислювальних блоків.

Відео-ОЗП – оперативно запам'ятовуючий пристрій, аналог загальної ОЗП, що відводиться лише під потреби ГП. Виконує функцію буфера, що містить у собі проміжні кадри, що не відображаються на екрані монітору. На даний час,

популярні такі типи пам’яті, як GDDR5, GDDR6 та HBM. Сучасні технології, також дозволяють при нестачі даної пам’яті задіяти частину загальної RAM.

З відео роз’ємів варто відзначити HDMI, DVI та DisplayPort. Зазвичай, вони комбінуються задля можливості приєднання декількох моніторів, зображення на кожному з котрих відображується окремо. Зазначимо, що HDMI має змогу передавати разом із зображенням звук, а DisplayPort дозволяє підключати до чотирьох різних пристроїв, наприклад, аудіо, USB-хаби тощо.

Підсумовуючи, маємо наступні характеристики:

- графічний процесор;
- покоління відеопам’яті;
- об’єм відеопам’яті;
- довжина відеокарти (для розміщення у корпусі).

1.2.6 Блок живлення

Блок живлення забезпечує вузли комп’ютера електричною енергією. До його функціоналу входить приведення змінного струму до постійного, його стабілізація та приведення до заданих значень напруги. Має у собі кулер, що забезпечує його охолодження. Останнім часом, з ростом загальної потужності систем та температур зсередини корпусу, розміщується у нижній частині системного блоку, забезпечуючи забір повітря ззовні та не заважаючи формуванню грамотних повітряних потоків всередині корпусу. Також блоки живлення мають захист від перенавантажень на збоїв у мережі. Часто до нього у пару докупляється окремо стабілізатор напруги або джерело безперебійного живлення.

Визначається потужністю (кількістю видаваємих Ватт) та ККД, що стандартизуються сертифікацією 80 PLUS. Потужність, записана у технічних характеристиках є потужністю на виході, тобто тою, що йде на забезпечення ПК, а від ККД залежить скільки блок живлення буде споживати електроенергії з мережі (таблиця 1.1).

Таблиця 1.1 – Сертифікація блоків живлення

Сертифікат	Навантаження від макс. потужності			
	10 %	20 %	50 %	100 %
80 PLUS	-	80 %	80 %	81 %
80 PLUS Bronze	-	81 %	85 %	81 %
80 PLUS Silver	-	85 %	89 %	85 %
80 PLUS Gold	-	88 %	92 %	88 %
80 PLUS Platinum	-	90 %	94 %	91 %
80 PLUS Titanium	90 %	94 %	96 %	91 %

Також бажано звернути увагу на інтерфейси підключення до материнської плати, процесора, відеокарти.

Єдиний елемент, що не накладає обмежень на інші, але має бути обраний відповідно до сумарного електроспоживання встановлених компонентів.

1.2.7 Системний блок

Функціональний елемент, що вміщує у собі усі компоненти, забезпечує їх кріплення, захищає від фізичного пошкодження зовнішніми чинниками, потрапляння пилу і бруду та надає можливість для монтажу додаткового охолодження. Виконує екрануючу функцію, оберагіючи користувача або навколишнє обладнання від електромагнітного випромінювання[1]. Зазвичай виробляється з металу, пластмаси, алюмінію, зустрічаються дизайнерські рішення з використанням прозорого пластику, загартованого скла, дерева.

Має відповідати форм-фактору материнської плати, враховувати висоту системи охолодження процесора та довжину відеокарти, не спричиняти конфлікту повітряних потоків.

1.3 Визначення рекомендаційної системи

Рекомендаційна система буде побудована на основі фільтрів, що додаються з кожним обраним компонентом системи та накладати обмеження на вибір наступних згідно до їх характеристик. Окрім того, буде додатково розроблена система, що за допомогою зміни кольору буде повідомляти користувача про те, що дані компоненти сумісні фізично, але у їх поєднанні немає сенсу, через велику різницю у потужності. Тобто, обираючи топовий процесор, користувач матиме змогу встановити найбільш бюджетну відеокарту, якщо, наприклад, йому не потрібна висока продуктивність при роботі з графікою, але такий вибір буде помічений системою як незбалансований. Це не заважатиме досвідченому користувачу, але попередить помилку необізнаного.

2 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

Для того, щоб розроблювана система відповідала усім сучасним критеріям стосовно такого роду програмного забезпечення було вирішено розглянути та проаналізувати існуючі рішення. Шляхом пошуку через мережу інтернет відібрано декілька найбільш популярних ресурсів, що пропонують схожий функціонал.

Аналіз даних рішень проводився відповідно до наступних критерій:

- наявність достатньої кількості фільтрів;
- захист від вибору незбалансованих та несумісних між собою комплектуючих;
- простий у розумінні та орієнтуванні інтерфейс;
- можливість створення акаунту;
- можливість збереження готових рішень.

2.1 Інтернет-магазин DigitalFury

Це онлайн платформа, що дозволяє сконфігурувати персональний комп’ютер самотужки або обрати існуючий та одразу його придбати. Враховуючи використання даного сайту як інтернет магазину, можна впевнитися, що фінальна ціна персонального комп’ютера буде дещо завищена через плату за надання послуг. Готові рішення гарно збалансовані, мають відгуки від власників, а отже можна вибрати найоптимальніше з них. Також на сайті є онлайн підтримка, що значно спрощує задачу з підбору персонального комп’ютера.

Сайт має перевантажений інтерфейс, що негативно впливає на можливість навігації. Не одразу зрозуміло, на яку кнопку тиснути, щоб відкрити ту чи іншу сторінку. Також, користувачеві пропонують почати вибір компонентів з платформи, тобто з чипсету та процесора. Обравши, наприклад, «Intel B360 Core i5 Конфигуратор», ми переходимо на сторінку де бачимо уже готовий ПК

побудований .на базі процесора Intel Core i5 9400F та материнської плати від невказаного виробника на чипсеті B360.

Користувач може або одразу придбати запропонований варіант, погодившись з іншими компонентами, що були підібрані автоматично, або почати внесення змін, кожне з яких змінюватиме фінальну ціну та потужність.

Вибір процесора одразу обмежується найактуальнішими моделями 9-го покоління, що унеможливлює його зміну на старіший, але й дешевший процесор 8-го, 7-го, чи іншого покоління, яке підходить до даного чипсету.

Список запропонованих відеокарт, що значно менший, ніж допустимий вибір для даної платформи. Відсутні такі варіанти, як наприклад, NVIDIA GeForce 1050 (звичайна та Ti), AMD Radeon RX 580 чи взагалі її відсутність з огляду на те, що даний компонент буде придбаний пізніше.

Також, у процесі огляду сайту було помічено логічну помилку, а саме можливість обрати пункт «Боксовый кулер процессора» коли виробник не передбачає наявність такого у комплекті.

2.2 Веб-платформа Telemart

Сайт має мінімалістичний вигляд, зрозуміло оформлений та не містить реклами. Присутні загальні фільтри, що потрібні для заохочення невимогливих покупців та допомагають сконцентруватися на основній характеристиці майбутнього ПК.

Фільтрування за призначенням хоч і виглядить цікаво, але не виконує гарно свої функції. Наприклад, фільтр «для роботи» не має сенсу, бо у роботі можуть використовуватися як прості комп’ютери, сконфігуровані для набору тексту, так і потужні рішення для обробки відео чи роботи з графікою. А модель, що може працювати з графічними матеріалами у більшості випадків буде гарним рішенням для геймерів. Більш логічним виглядає блок фільтра «для ігор» - дуже часто, користувач має бажання грати в певну гру, під яку і збирає собі персональний комп’ютер.

Вікно вибору процесора містить у собі окремі фільтри, що застосовуються лише для процесорів. Такі самі вікна створені для кожного іншого компонента. Більш того, є також блоки для периферії та аксесуарів.

Також, на сайті є вже описана проблема, що проявляється й на минулому рішенні (DigitalFury), а саме можливість вибору занадто потужного, у порівнянні з іншими, компоненту. Наприклад, без повідомлень про дисбаланс зборки обирається материнська плата на AMD X570 чипсеті, що призначений для надпотужних систем, як мінімум з AMD Ryzen 7 та вище. Тим часом, у нашому виборі знаходиться процесор початкового класу AMD Ryzen 3. Подібна ситуація й з вибором відеокарти на рисунку – даний процесор не розкриє повністю потенціал обраної, Nvidia GeForce 2080 Ti, відеокарти, що призведе до її простою, а відповідно, користувач переплатить за потенціал потужності, що він не отримає.

2.3 Інтернет-магазин InTeam

Інтернет-магазин InTeam також має свій власний конфігуратор, що створений для заохочення клієнтів до покупок на їх ресурсі. Головна сторінка сайту виконана у мінімалістичному інтерфейсі з мінімумом картинок та повністю без реклами, що не відволікає користувача від елементів управління.

У вигляді зрозумілої схеми представлено ПК, є додаткові блоки для вибору опціональних компонентів, таких як система охолодження та периферія. Присутні декілька основних фільтрів для вибору загального типу ПК – ігровий, домашній чи офісний, хоча не зрозуміло, яка різниця між домашнім та офісним типом, бо задачі ставляться перед такими комп’ютерами однакові. Це, як правило, робота з текстовими документами та базові мультимедійні властивості. Сайт пропонує одразу придбати всі компоненти, так як він сам є їх постачальником.

Не є зрозумілою можливість вибору трьох планок оперативної пам’яті, бо комплектуючі від різних виробників можуть не працювати коректно одна з одною. Натомість вибір двох відеокарт, що мають можливість співпрацювати,

тим самим підвищуючи загальну продуктивність ПК, використовуючи такі інтерфейси, як SLI та CrossFire розроблені відповідно Nvidia та AMD відсутній.

Відсутні такі варіанти фільтрації, як фільтрація за наявними інтерфейсами для виведення графічних даних (HDMI, VGA, DVI) та підключення постійної пам'яті, будь-то жорсткий диск, що працює використовуючи різні покоління SATA інтерфейсу, чи твердотільний накопичувач, що може підтримувати nVME протокол та M.2 інтерфейс. Відсутні також пункти з сокетом та чипсетом.

Ситуація з фільтрами аналогічна до вибору материнської плати. Також існує можливість обрати сокет, що не підходить до обраної материнської плати, що робить дану систему недієздатною та такою, що може завдати шкоду недосвідченому користувачу. Виходячи з цього, робимо висновок, що дане готове рішення має більше недоліків ніж переваг (зрозумілого та мінімалістичного, а отже не перенавантаженого інтерфейсу) і є, скоріш, прикладом того, чого потрібно уникнути при розробці власного рішення.

2.4 Гра PC Creator

Найцікавішим рішенням є гра, що нещодавно набула великої популярності, - PC Creator. Хоча дане рішення не є спеціалізованою програмою для спрощення процесу проектування ПК, вона є найкращим примірником орієнтованості на користувача, містить у собі безліч різноманітних підказок та інструкцій.

Будучи грою, має у собі ігрові елементи, такі як віртуальні гроші та досвід та базується на розвитку персонажа шляхом виконання певних замовлень. Даний процес якомога найкраще та найцікавіше знайомить користувачів з будовою ПК та правилами його проектування.

У процесі збірки ПК на якому видно, що назви фірм, що виготовляють комплектуючі дещо змінені, але не настільки, щоб вгадати було неможливо. Досвідчений користувач легко зрозуміє, що RMD Ryzen 2700X є AMD Ryzen 7 другого покоління, та варто відмітити, що перша цифра (7 в даному випадку) безпосередньо вказує на рівень потужності даного ЦП, тож не вказання її можна

віднести до недоліків. Також відсутній обсяг відеопам’яті у відеокарти, частота у оперативної пам’яті, розгорнуті характеристики процесору, постійної пам’яті.

Натомість цікавий, простий та яскравий інтерфейс позитивно впливає на користувацький досвід.

Після виконання завдання можна отримати оцінку збірки та, таким чином, навчатися та покращувати свої навички у проектуванні. Тож, підсумовуючи вищесказане, даний додаток можна вважати не більш, як гарною інструкцією для тих, хто тільки почав розбиратися в комп’ютерній техніці.

2.4 Висновки

Перевагами розглянутих систем є те, що усі рішення є сайтами, тож вони безкоштовні та загальнодоступні, часто існують як додатки до інтернет магазинів. Більшість з них будує свій інтерфейс у вигляді списку, але є цікавий варіант, що являє собою малюнок комп’ютера та комірки, якими обігруються майбутні складові персонального комп’ютера. Такий підхід дозволяє на інтуїтивному рівні навчати користувача внутрішній структурі та уникати нав’язування порядку дій, даючи змогу починати проектування ПК з любої частини .

Щодо недоліків, то це перевантаженість інтерфейсу, що відволікає від основної задачі, та скупий вибір фільтрів, що не дозволяє адекватно фільтрувати кожен з компонентів за рядом ознак, що є важливими у збиранні збалансованого персонального комп’ютера. Найбільшим недоліком цих рішень є відсутність системи, що надає рекомендації з сумісності та доцільності використання певних компонентів разом. Найяскравіший приклад – можливість обрати у пару до процесора початкового рівня найпотужнішу на даний час відеокарту. У такій ситуації процесор просто не зможе повністю забезпечити відеокарту роботою і вона буде простоювати. Це є недоцільне використання ресурсів, уникнення чого дозволить зекономити користувачу кошти, отримавши систему з не меншою вихідною продуктивністю. Також було виявлено можливість поєднати несумісні одне з одним компоненти ПК.

Отже, розроблюваний додаток має бути кросплатформеним з інтуїтивно зрозумілим інтерфейсом та великим вибором фільтрів, що дозволить максимально точно корегувати поведінку користувача та максимально ненав’язливо направляти його шляхом підбору сумісних одне з одним компонентів. Враховуючи орієнтованість додатку на недосвідчених користувачів є сенс на ранньому етапі розробки скрити фільтри взагалі, зробивши їх невидимими, а проводити фільтрацію безпосередньо у коді, автоматично, після вибору кожного комплектуючого. Також, необхідна система, що дозволить попереджувати користувача про те, що хоча певні компоненти і є сумісними у фізичному плані, але вони відносяться до різних категорій потужності, тож є сенс переглянути свій вибір.

3 ОПИС РОЗРОБЛЮВАНОЇ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ

3.1 Інтерфейс користувача

Аналіз існуючих рішень дає уявлення інтерфейсу, що необхідно реалізувати у автоматизованій системі надання рекомендацій. Інтерфейс має бути продуманим та інтуїтивно зрозумілим, своїм виглядом спонукати користувача до розуміння архітектури персонального комп'ютера. Найбільш вдалим на мою думку є інтерфейс InTeam, через його мінімалізм та схематичність.

Отже, було сформовано наступні вимоги:

– головна сторінка з схематичним інтерфейсом збору ПК та шістьома комірками;

– кнопка аналіз зборки та чек-бокс для відображення доданих користувачами компонентів;

– сторінка авторизації та додавання компоненту з мінімально необхідною кількістю текстових полей;

– сторінка перегляду збережених комп'ютерів у вигляді плиток;

– сторінка адміністратора з можливістю видаляти юзерів та додані ними компоненти.

3.2 Функціональні можливості

Дана система являє собою desktop програму, що має можливість збирати комп'ютер, зберігати його у своєму профілі та аналізувати збалансованість збірки. Отже повинна мати наступних п'ять вікон:

- сторінка збору;
- сторінка додавання відсутніх у базі даних комплектуючих;
- сторінка авторизації;
- сторінка перегляду збережених збірок.

Кожна сторінка має власний набір функціональних дій, що надає користувачеві певні можливості.

Сторінка збору дає змогу обирати необхідні компоненти з бази даних, переглядати їх характеристики, аналізувати збалансованість, зберігати готове рішення у своєму профілі.

Сторінка додавання відсутніх у базі даних комплектуючих підтримує актуальність існуючої бази даних шляхом надання користувачам прав на внесення нових компонентів до існуючої БД. Внесені користувачами дані, помічаються спеціальним флагом, що дозволяє користувачеві обирати, чи хоче він звертати свою увагу тільки на перевірену розробником інформацію, чи допускає використання знову доданих елементів.

Сторінка авторизації існує для реєстрації нових користувачів, якщо в них є бажання зберігати свої комп'ютери, або авторизації, якщо вони хочуть передивитися вже збережені.

Сторінка перегляду збережених збірок містить у собі персональний список комп'ютерів кожного користувача з можливістю видалення або редагування їх.

3.3 .NET Core як інструмент розробки

Враховуючи, що усі готові рішення існують як сайт, було вирішено розробити власний додаток кросплатформеним, використовуючи для цього можливості платформи .NET Core. Вона розроблена та підтримується компанією Microsoft та є сумісною з наступними операційними системами – Windows, Linux, MacOS. Так як це вільно-поширювана платформа, користувачі можуть використовувати її безкоштовно у некомерційних цілях. Великою перевагою є те, що дана платформа модульна, тобто, дозволяє використовувати тільки необхідні модулі у програмі. Це значно зменшує кількість вільного місця, що потребує завершений проект.

Також, ця платформа дозволяє використовувати такий зручний фреймворк для роботи з базою даних як Entity Framework. Підтримуваними мовами розробки є об'єктно орієнтовані мови C#, Visual Basic .NET та функціональна F#.

3.4 Середовище та мова розробки

Найпопулярнішим середовищем розробки на платформі .NET Core є Microsoft Visual Studio, що дозволяє розроблювати як прості консольні додатки так і додатки з графічним інтерфейсом. Підтримує такі технології створення графічних інтерфейсів як Windows Forms, WPF (Windows Presentation Foundation), UWP (Universal Windows Platform)[2].

Розробка буде проводитись з використанням об’єктно орієнтовної мови C#[3]. Вона є строго типізованою, тому проста та безпечна. Будучи розробленою у 1998-2001 роках, вона залишається актуальною через підтримку Microsoft. Компанія постійно випускає оновлення, забезпечуючи стабільність роботи та появу нового функціоналу.

Для реалізації бази даних, використано мову структурованих запитів SQL та середовище MSSMS (Microsoft SQL Server Management Studio). Дана мова призначена для створення, модифікації та керування даними у реляційній базі даних.

3.5 Entity Framework

Розроблений корпорацією Microsoft у 2008 році як основний засіб взаємодії між аплікаціями .NET та реляційними базами даних. Інструмент, що значно спрощує проектування та співставлення об’єктів у програмному забезпеченні з таблицями та стовпиками реляційної бази даних[4].

Будучи ORM-фреймворком обробляє створення з’єднання з базою даних та виконання команд, а також результати запитів та автоматичне співставлення цих результатів у якості об’єктів додатку. Також допомагає відстежувати зміни об’єктів додатку та може зберігати ці зміни у базі даних. Підвищує продуктивність скорочуючи задачі зі збереження даних, що використовується у додатку. Підтримує написання запитів з використанням LINQ.

3.6 MS Test

Фреймворк юніт-тестування від компанії Microsoft, що за замовчуванням міститься в деяких версіях Visual Studio починаючи з VS2005. Дозволяє писати юніт-тести безпосередньо у середовищі розробки VS та впроваджує систему атрибутів для позначення класів, що тестуються та тих, що безпосередньо виконують тестування.

3.7 Хостинг VCS GitHub

GitHub – це веб платформа, що існує задля хостингу проектів, що працюють з різними системами контролю версій, як Git, Mercurial, SVN. Даний проект використовує дану платформу з VCS Git, що забезпечує можливість відстеження змін у проєкті, відкатувати їх та визначити дату та обсяг внесення.

Надає наступні можливості:

- можливість скрити репозиторій;
- дозволяє вести облік змін у вигляді дерева;
- створення окремих гілок для різних змін;
- ведення проєкту сумісно з іншими розробниками.

4 ВИКОРИСТАНІ ШАБЛОНИ ТА ПІДХОДИ ПРОЕКТУВАННЯ

4.1 Об’єктно орієнтовне програмування

Об’єктно орієнтовне програмування – це методологія програмування, що базується на представлені програми у вигляді сукупності об’єктів, кожен з яких описується певним класом, що утворює ієрархію наслідування.

Ідеологія ООП базується на моделюванні інформаційних об’єктів, що вирішують задачу структурного програмування з точки зору керування[5]. Це суттєво покращує керованість процесу моделювання що, у свою чергу, дуже важливо при реалізації великих проєктів.

Керованість в системах з певною ієрархією передбачає мінімізацію надмірності даних (аналогічно до нормалізації) та їх цілісність, бо те, що зручно керується – простіше розуміється, що є важним аспектом при довготривалій підтримці проєкту або сумісній розробці[6].

Можна виділити чотири принципи структурування, що пов’язані з різними аспектами базового розуміння предметної задачі:

– абстракція – контекстне розуміння предмету, що формалізується у вигляді класу. Виділити у предметі, що моделюється, того важливого, що необхідно для рішення поставленої задачі

– інкапсуляція – організовує ієрархічність керування, для відділення, наприклад, простої команди «зроби те» від уточнення як саме це робити, через різні рівні керування

– наслідування – використовується для зручної та безпечної організації споріднених понять. Таким чином стає можливим враховувати тільки ті зміни, що мають бути присутні на даному ієрархічному кроці розробки, не дублюючи ті що було розроблено на минулих кроках;

– поліморфізм – потрібен для відокремлення або, навпаки, поєднання управління в тих чи інших частинах коду.

4.2 SOLID принципи

Принципи, що носять рекомендаційний характер для розробників програмного забезпечення та описують правила побудови слабо зв’язного та простого до розширення коду. Абревіатура SOLID складена з перших літер кожного принципу, а саме:

– S – Single Responsibility Principle – принцип єдиного обов’язка, тобто кожна функціональна одиниця коду (клас, метод тощо) повинна мати лише одну відповідальність та виконувати лише одну функцію. Якщо клас виконує декілька різних функцій, то це є підставою для його зміни відповідно до даного принципу;

– Open/Closed Principle – принцип відкритості/закритості, тобто функціональні одиниці мають бути відкритими для розширення, але закритими для зміни. Принцип рекомендує проектувати систему таким чином, щоб додавання нового функціоналу не тягнуло за собою зміни у вже існуючому;

– L – Liskov Substitution Principle – принцип підстановки Лісков описує як правильно будувати ієрархію наслідування. Спрощує використання поліморфізму через можливість підстановки замість базового типу дочірнього. При грамотному застосуванні дозволяє використання інтерфейсів замість класів;

– I – Interface Segregation Principle – описує правила створення інтерфейсів задля уникнення їх розростання, що у народі називається «жирний інтерфейс». Це значно ускладнює роботу з такими інтерфейсами, бо зобов’язує описувати інтерфейси разом з методами, що можуть не використовуватись у певних класів. Рекомендується розділяти такі інтерфейси на декілька мінімальних логічно-завершених;

– D – Dependency Injection Principle – описує створення слабо зв’язних сутностей, відкритих до модифікації, тестування та підтримки. Найпопулярніше формулювання звучить так: модулі верхнього рівня не мають залежити від модулів нижнього рівня, усі повинні залежати від абстракцій, абстракції не залежать від деталей, а деталі залежать від абстракцій.

4.3 Code First

Наразі в EntityFramework існують та активно використовуються два підходи до розробки бази даних та програмної частини додатку (рисунок 4.1) – «Code First» та «Database First». В залежності від значення тої чи іншої частини обирається відповідний підхід.

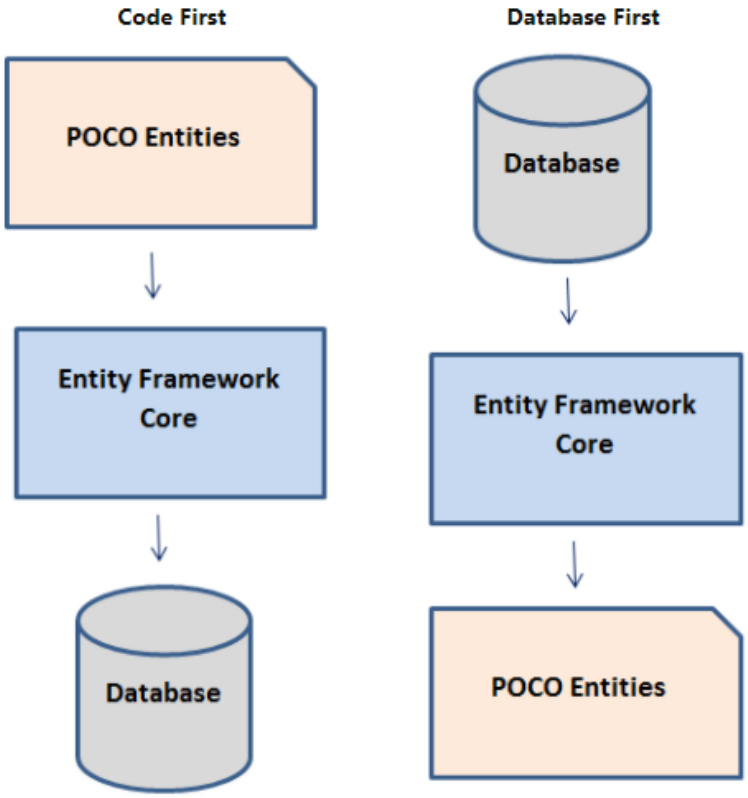

Рисунок 4.1 – Підходи проектування бази даних та програмної частини

Підхід, що зветься Code First (спочатку код) передбачає мінімальну участь в проектуванні сутностей бази даних програміста. Він просто пише код, інше робить за нього Entity Framework та Visual Studio[7].

Він підходить у випадках якщо головне в проекті – бізнес логіка, а база даних – це спосіб зберігання даних. Або у випадках, коли проект вже написаний, але в якості джерел даних використані списки, масиви, колекції. Code First дозволяє з мінімальними зусиллями змінити проект з використанням баз даних в якості джерел даних замість стандартних колекцій .NET.

Переваги підходу CodeFirst, за які його було обрано до проекту:

– спрощення реалізації бази даних шляхом знищення необхідності до ручного співставлення відповідностей моделі додатку та БД;

– можливість застосування міграцій (план переходу БД від старої схеми до нової), що полегшує внесення змін до існуючої бази даних при зміні моделі даних у додатку;

– створення контексту даних, як підхід проектування, що забезпечує взаємодію між класами-моделями та однойменними таблицями у БД.

4.4 Шаблон проектування MVVM

MVVM (Model-View-ViewModel) – це похідний від MVC (Model-View-Control) шаблон проектування, що застосовується для відокремлення моделі та її представлення при проектуванні архітектури додатку[8]. Зменшує зв’язність коду, що полегшує подальше розширення та підтримку існуючого додатку.

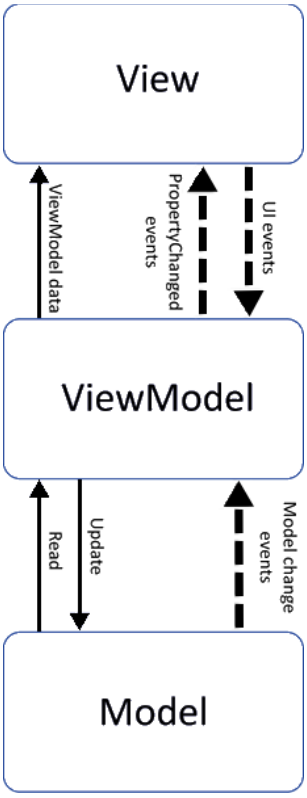

Рисунок 4.2 – Візуальне представлення патерну MVVM

В основі лежить створення трьох слабо зв’язних частин (рисунок 4.2), кожна з яких має свою зону відповідальності:

– модель (Model) подібно до класичного MVC є відображенням якогось функціонального об’єкту;

– вигляд/Відображення (View) є графічним інтерфейсом, що відображує передані йому дані та надає користувачеві інтерфейс взаємодії з програмою;

– модель Вигляду (ViewModel) є одночасно абстракцією вигляду з одного боку, та з іншого надає обгортку даних Моделі. Тобто призначенням ViewModel є реалізація зв’язку між моделлю та її відображенням, відслідковування змін, внесених користувачем та обробка логіки роботи View – відпрацювання механізму команд.

4.5 Патерн Singletone

Одинак – породжуючий патерн, який гарантує, що у класу є тільки один екземпляр, і надає до нього глобальну точку доступу[9].

Для деяких класів важливо, щоб існував тільки один екземпляр. Наприклад, хоча в системі може бути багато принтерів, але можливий лише один спулер. Повинні бути тільки одна файлова система і єдиний віконний менеджер. У цифровому фільтрі може перебувати тільки один аналого-цифровий перетворювач (АЦП). Бухгалтерська система обслуговує тільки одну компанію.

Найпростіше коли глобальна змінна дає доступ до об'єкта, але не забороняє інстанціювати клас в декількох примірниках. Більш вдале рішення – сам клас контролює те, що у нього є тільки один екземпляр, може заборонити створення додаткових примірників, перехоплюючи запити на створення нових об'єктів, і він же здатний надати доступ до свого екземпляру.

4.6 Шаблонний метод

Даний патерн є поведінковим, визначає скелет алгоритму делегуючи відповідальність за певний функціонал з класів на їх підкласи[10,11]. Таким

чином є можливим перевизначення деяких кроків алгоритму у дочірніх класах без зміни загальної структури.

Таким чином, на начальних етапах проектування вирішується які кроки алгоритму є незмінними, а які змінними. Стандартні кроки алгоритму формуються та реалізуються у базовому класі. Змінні кроки надаються клієнтом компонента в певних похідних класах.

Найчастіше даний патерн використовується в каркасах додатків (фреймворках), де кожен каркас реалізує незмінні частини архітектури предметної області та визначає ті частини, що їх можна налаштовувати клієнту за такої необхідності.

4.7 Патерн memento

Даний шаблон проектування є поведінковим. Дозволяє без порушення принципів ООП, а саме інкапсуляції, зберігати та, за потреби, повертати стан будь-якого об’єкта, що його реалізовує без виносу стану назовні. Найчастіше використовується при реалізації контрольних точок і механізмів відкату[9], що дозволяють скасовувати хибну операцію або відновляти стан додатку після помилки.

4.8 Патерн Команда

Команда – поведінковий патерн, що інкапсулює запит як об'єкт, дозволяючи тим самим задавати параметри клієнтів для обробки відповідних запитів, ставити запити в чергу або протоколювати їх, а також підтримувати скасування операцій. Відомий також під назвами Action (дія), Transaction (транзакція).

Іноді необхідно надсилати об'єкти запитів, нічого не знаючи про те, виконання якої операції запрошено і хто є одержувачем[9]. Наприклад, в бібліотеках для побудови призначених для користувача інтерфейсів зустрічаються такі об'єкти, як кнопки і меню, які надсилають запит у відповідь на дію користувача. Але в саму бібліотеку не закладена можливість обробляти

цей запит, так як тільки додаток, що використовує її, має інформацію про те, що слід зробити. Проектувальник бібліотеки не володіє жодною інформацією про одержувача запиту і про те, які операції той повинен виконати.

Патерн «команда» дозволяє бібліотечним об'єктам відправляти запити невідомим об'єктам докладання, перетворивши сам запит в об'єкт. Цей об'єкт можна зберігати і передавати, як і будь-який інший. В основі описаного патерну лежить абстрактний клас Command, в якому оголошено інтерфейс для виконання операцій. У простій своїй формі цей інтерфейс складається з однієї абстрактної операції Execute. Конкретні підкласи Command визначають пару «Одержувач-дія», зберігаючи одержувача в змінній примірника, і реалізують операцію Execute, так щоб вона надсилала запит. У одержувача є інформація, необхідна для виконання запиту.

У даному додатку для спрощення реалізації було використано готову бібліотеку з NuGet менеджеру, а саме MvvmLightLibs та їх реалізація патерну Command – RelayCommand.

5 АНАЛІЗ РОЗРОБЛЕНОЇ СИСТЕМИ

5.1 Структура проекту

Відповідно до рішень, прийнятих у пункті 3, при розробці було застосовано шаблон проектування MVVM, створено директорії під Моделі (Models), Відображення (Views) та МоделіВідображення (ViewModels). За допомогою механізму прив’язки (Binding), що реалізований у технології WPF було скоординовано дії відображень щодо відкриття/закриття певних вікон та коректної обробки представлених даних.

Структура додатку складається з наступних компонентів:

– класи для роботи з базою даних: «DatabaseLists» та «DBContext» у директорії «Database»;

– окрема директорія для картинок «images»;

– директорія, що зберігає історію міграцій, що були застосовані до БД, папка «Migrations»;

– директорії, що зберігають окремо одне від одного функціональні елементи MVVM моделі, а саме:

- директорія Views з відображеннями у вигляді UserControl’ів;
- директорія Models з моделями;
- директорія ViewModels для зберігання відображень моделей.

Система має 8 класів та 2 інтерфейси що показано на діаграмі класів ІА61.310БАК.005 Д3. До класів віднесено усі комплектуючі, збірна сутність «Computer», що представляє собою клас, зберігаючий конфігурацію, та користувач «User», що має змогу аутентифікації та перегляду збережених варіантів. Інтерфейс IMyComponent описують усі без виключення компоненти системи базовими їх характеристиками, а саме ідентифікатором, іменем, класом потужності та роком випуску. Інтерфейс IPowerConsumptor описують лише ті класи, яким потрібно враховувати своє енергоспоживання для подальшого вибору блоку живлення.

Для реалізації зв’язку між класами було використано тип зв’язку один до багатьох, що означає, що користувач може мати багато комп’ютерів, але кожен

комп’ютер може мати лише одного користувача та, що один комп’ютер може містить у собі лише один елемент будь-якого компоненту у той час, як цей самий компонент може являтися частиною багатьох комп’ютерів. Це наглядно представлено на діаграмі ІА61.310БАК.005 Д4, що є діаграмою структури бази даних.

5.2 Опис роботи додатку

Наразі у додатку реалізовано два вікна:

– MainWindow, що містить у собі посилання на BuildingUC (рисунок 5.1), AuthorizationUC (рисунок 5.3), RegistrationUC (рисунок 5.4), SavedPCUC (рисунок 5.8) та дозволяє користувачу збирати комп’ютер, авторизуватися, реєструватися та переглядати збережені комп’ютери відповідно. Також має кнопки для виклику вікна додавання компоненту та управління авторизацією;

– AddComponentWindow, що містить у собі декілька UserControl’ів – по одному на кожний компонент, кожен з яких надає змогу користувачеві додати свій компонент, якого ще немає у базі даних. Такий UserControl має декілька полів, для введення характеристик компоненту та кнопку «Add».

Рисунок 5.1 – Інтерфейс BuildingUC

Кожне з вікон є хостингом для певних UserControl’ів, що значно зменшує кількість вікон, які не відкриваються у результаті взаємодії з програмою, а просто поміщаються в спеціальний елемент на формі.

Робота з системою починається з вікна конструктора (рисунок 5.1), що має шість плиток для вибору компонентів та перегляду їх характеристик, кнопка «+» ліворуч кожної плитки для додавання компонентів, кнопка «Login» для відкриття вікна AuthorizationUC (рисунок 5.3), кнопка «Check» для запуску перевірки відповідності компонентів одне одному та флаг «Show user-added» для ввімкнення/вимкнення відображення доданих користувачами компонентів у випадаючих списках. Таким чином, користувач може фільтрувати компоненти, що йому рекомендує додаток.

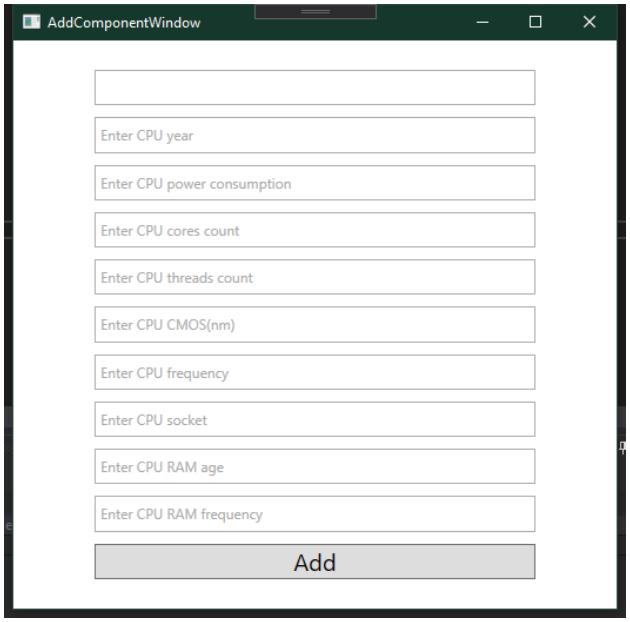

Рисунок 5.2 – Вікно додавання компоненту на прикладі додавання CPU

На рисунку 5.2 бачимо вікно додавання CPU, що робиться шляхом заповнення усіх полів релевантними значеннями та натисканням кнопки «Add». У разі невідповідності даних у полі типу змінної (наприклад введення тексту в поле частоти), воно набуває червоної рамки та робить кнопку «Add» неактивною перешкоджаючи таким чином виникненню помилок через невідповідність типу даних. Подібний UC реалізовано й для інших компонентів з урахуванням їх характеристик.

З рисунку 5.3 бачимо інтерфейс вікна авторизації, що являє собою два поля для заповнення – «login» та «password», та дві кнопки – «login» і «create account» що виконують вхід до системи та відкривають вікно реєстрації (рисунок 5.6) відповідно. У разі некоректного вводу логіну чи пароля буде показано відповідне вікно з проханням повторити введення та пропозицію зареєструватися.

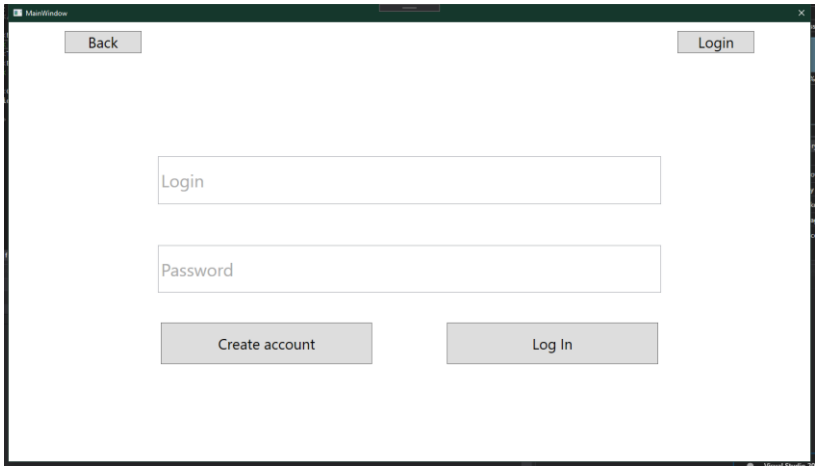

Рисунок 5.3 – AuthorizationUC

На рисунку 5.4 представлено UC реєстрації, він також розміщується в основному вікні (MainWindow). Має три TextBox'и для вводу логіна, пароля та повторного вводу пароля, задля уникнення помилок на етапі реєстрації. Одна кнопка «Create account», що стає активною при заповненні полів та відповідності паролів одне одному. При наявності акаунта з введеним логіном показується MessageBox з відповідним текстом.

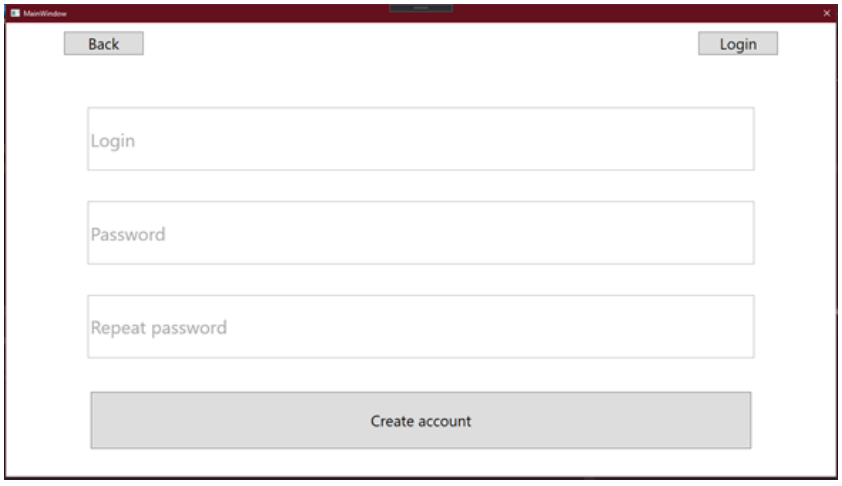

Рисунок 5.4 – RegistrationUC

На рисунку 5.5 представлено UC, на якому містяться плитки зі збереженими користувачем комп’ютерами. Дане вікно отримує можливість бути відкритим після авторизації користувача при наявності у нього збережених ПК. За дану дію відповідає кнопка «Saves», що розміщена на керуючій панелі (рисунок 5.10) вікна MainWindow.

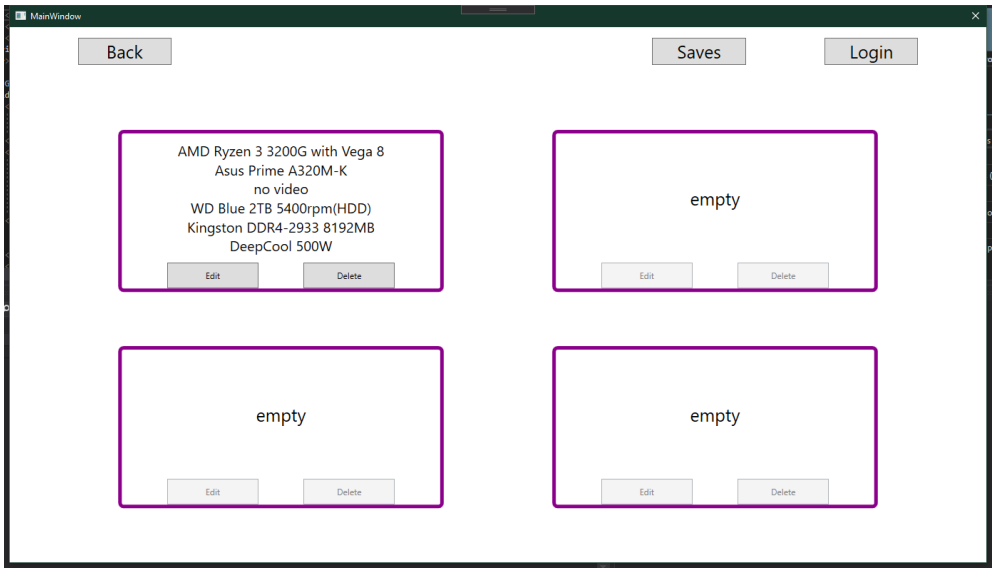

Рисунок 5.5 – SavedPCUC

Керуюча панель, що займає верхню частину MainWindow, до авторизації, має 3 кнопки – «Back», «Saves», «Login». Кнопка «Back» працює за допомогою шаблону «Memento» та повертає до вікна попередній UC. Після авторизації дана панель змінюється – з’являється привітання з логіном теперішнього користувача та кнопка «Saves» для завантаження SavedPCUC (рисунок 5.5).

Даний UC містить у собі відображення у вигляді плиток раніше збережених користувачем ПК. Кожна плитка містить дві кнопки «Edit» та «Delete». Перша – відкриває збірку у BuildingUC і дозволяє таким чином зробити потрібні зміни, друга – видаляє збережену збірку.

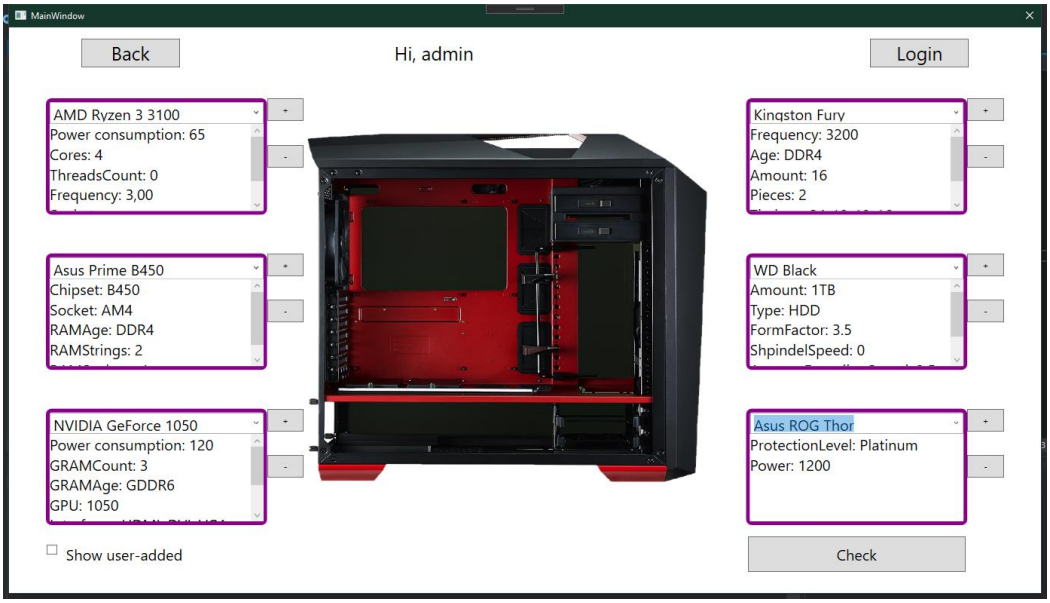

Рисунок 5.6 – BuildingUC при авторизації адміна

На рисунку 5.6 представлено UC адміністратора, що відкривається після введення у форму авторизації логіну та паролю адміністратора. Він має змогу переглядати зареєстрованих користувачів та додані ними компоненти і видаляти їх за необхідності кнопкою «Remove».

5.3 Розробка бази даних

База даних спроектована за допомогою Entity Framework. Містить у собі 8 основних таблиць та одну з міграціями. У таблиці 5.1 відображено призначення кожної з них – сутності, що зберігаються та їх поля.

Таблиця 5.1 – Призначення таблиць бази даних

Назва таблиці	Опис
Users	Таблиця зберігає дані про користувачів, зокрема їх логін (Login) та пароль (Password)
Computers	Містить дані про збережені конфігурації комп’ютерів та

	користувачів, що їх сконфігурували. Поля: – Cpu_Id; – GraphicCard_Id; – Memory_Id; – Motherboard_Id; – PowerSupply_Id; – RAM_Id; – User_Id. Є зовнішніми ключами, що посилаються на інші таблиці.
CPUs	Таблиця з відкритими до вибору користувачів процесорами, містить у собі: – назву (Name); – клас потужності (PowerClass); – рік випуску (Year); – споживання електроенергії (PowerConsumption); – к-сть ядер (CoresCount); – частота (Frequency); – чипсет (Chipset); – покоління оперативної пам’яті (RamAge); – частота оперативної пам’яті (RamFrequency); – флаг, чи є компонент доданий звичайним користувачем чи адміністратором (AddedByUser); – кількість потоків (ThreadsCount); – техпроцес у нм (CMOS).

GraphicCards	Перелік відеокарт та інформація про них: – назва (Name); – клас потужності (PowerClass); – рік випуску (Year); – споживання електроенергії (PowerConsumption); – об’єм відеопам’яті (GRAMCount); – покоління відеопам’яті (GRAMAge); – відеопроцесор (GPU); – інтерфейси підключення (Interfaces); – роз’єм підключення додаткового живлення (PowerType); – версія DirectX (DirectX); – флаг, чи є компонент доданий звичайним користувачем чи адміністратором (AddedByUser).
Memories	Таблиця зберігає дані про постійну пам’ять з такими полями: – назва (Name); – клас потужності (PowerClass); – рік випуску (Year); – об’єм пам’яті (Amount); – тип HDD чи SSD (Type); – інтерфейс підключення (ConnectionInterface); – форм-фактор (FormFactor);

	– швидкість шпинделя (ShpindelSpeed); – середня швидкість пошуку (AverageFoundingSpeed); – флаг, чи є компонент доданий звичайним користувачем чи адміністратором (AddedByUser).
Motherboards	Містить у собі дані про материнські плати, а саме: – назву (Name); – клас потужності (PowerClass); – рік випуску (Year); – чипсет (Chipset); – сокет (Socket); – покоління оперативної пам’яті (RAMAge); – к-сть потоків оперативної пам’яті (RAMStrings); – к-сть роз’ємів під оперативну пам’ять (RAMSockets); – флаг, чи є компонент доданий звичайним користувачем чи адміністратором (AddedByUser).
PowerSupplies	Перелік можливих до встановлення блоків живлення з такою інформацією: – назва (Name); – клас потужності (PowerClass); – рік випуску (Year); – рівень захисту від перенавантажень у мережі (ProtectionLevel);

	– потужність (Power); – флаг, чи є компонент доданий звичайним користувачем чи адміністратором (AddedByUser).
RAMs	Таблиця зберігає дані про оперативну пам’ять, містить у собі інформацію про: – назву (Name); – клас потужності (PowerClass); – рік випуску (Year); – частоту (Frequency); – покоління (Age); – об’єм (Amount); – к-сть планок (Pieces); – таймінги (Timings); – флаг, чи є компонент доданий звичайним користувачем чи адміністратором (AddedByUser).

5.4 Подальша підтримка та розширення функціоналу

Під час проектування були виконані усі завдання, що поставлені перед додатком на даний момент. Завдяки дотриманню шаблона MVVM подальші зміни можуть бути легко інтегровані через слабко зв’язаний код. Тобто, легко впровадити зміни, наприклад, бізнес логіки, бо вона розташована окремо від представлення та навпаки, є змога легко змінити відображення, через його слабку зв’язність з моделями.

Також має сенс впровадження декількох вікон з максимальною кількістю можливих фільтрів для кожного з компонентів з автоматичним виставленням позначок у фільтри відносно вже обраних компонентів. Обов’язково надати

можливість зняття їх задля розширення кола потенційних користувачів обізнаними у технологіях людьми, що потребують від даного додатку лише зручного менеджера.

Надалі є сенс додати функціональність, що дозволить користувачу обирати декілька певних компонентів одразу, таких як відеокарта чи постійна пам’ять. З впровадженням такими компаніями як AMD та Nvidia технологій CrossFireX та, відповідно SLI, таке рішення є досить коштовним, але й досить перспективним, дозволяючи підняти графічну потужність комп’ютера вище двома топовими відеокартами, або трохи додати йому конкурентоспроможності шляхом подвоєння бюджетних відеокарт. Щодо постійної пам’яті, наразі існує технологія створення RAID-масивів, що застосовує декілька дисків для їх об’єднання та підвищення надійності і продуктивності накопичувачів.

Є необхідність інтеграції даного застосунку до інтернет-платформи з продажу комп’ютерних комплектуючих задля отримання доступу до бази даних, що постійно оновлюється. Таким чином буде отримано актуальні ціни на комплектуючі та не зайвим стане функціонал, що дозволить напряму робити замовлення з додатку, минаючи перенесення збірки з додатку до інтернет-магазину.

6 ТЕСТУВАННЯ РОЗРОБЛЕНОГО РІШЕННЯ

Тестування є одним з основних етапів циклу розробки програмного забезпечення. Це процес перевірки відповідності розробленої системи до вимог, що висуваються у процесі проектування даної системи та реально реалізованого функціоналу. Зазвичай кількість можливих тестів, що можуть бути складені до певного програмного продукту прямує до нескінченності, тож розроблюється стратегія, що покриває більшість проблемних ділянок відповідно до наявного часу та ресурсів.

Тестування ПЗ може надавати об’єктивну, незалежну інформацію про якість ПЗ, ризики відмови, як для користувачів, так і для замовників[12]. У проекті було використано наступні види тестування ПЗ:

- за ступенем автоматизації:
  - ручне тестування (manual testing);
  - автоматизоване тестування (automated testing);
- за ступенем підготовленості до тестування – інтуїтивне тестування, або ad hoc – тестування без плану та тестової документації, базується на методиці передбачення помилок на власному досвіді тестувальника або розробника. Найліпший у даному випадку варіант, через те, що додаток розробляється та тестується однією людиною, не є надто великим та потребує тестування лише базових функцій;
- за ступенем ізольованості компонентів – модульне тестування (unit testing).

6.1 Концепція тестування TDD

У процесі розробки було застосовано концепцію розробки через тестування TDD (Test-Driven-Development). Вона представляє процес застосування unit-тестів, при якому спочатку пишуться самі тести, а потім програмний код, достатній для їх проходження.

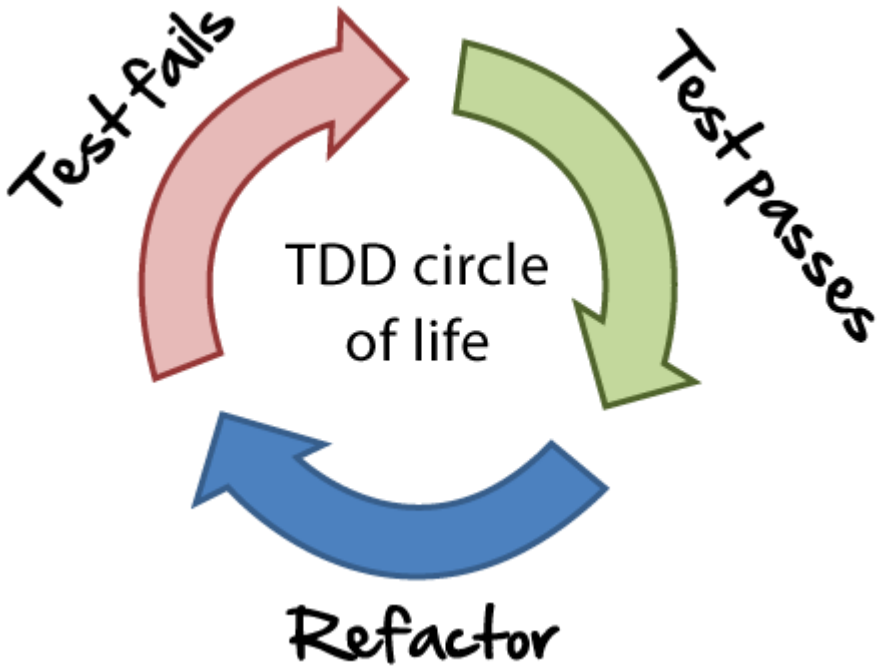

Рисунок 6.1 – Цикл TDD[11]

Дозволяє значно знизити кількість потенційних помилок у додатку. Створюючи тести перед написанням коду ми одразу визначаємо поведінку майбутніх компонентів не заважаючи процесу конкретною реалізацією компонентів, що підлягають тестуванню[13]. Таким чином готові тести є чимось на кшталт шаблону та дозволяють одразу описати API майбутніх компонентів.

Дана концепція має наступну послідовність дій:

– написання unit-тестів;

– їх запуск, задля визначення тестів як не пройдених;

– написання коду, достатнього для проходження тестів;

– запуск тестів, для переконання у правильності написаного коду та подальше його доповнення під контролем вже реалізованих тестів.

6.2 Реалізовані автоматичні unit-тести

Під час розробки було реалізовано наступні unit-тести представлені у таблицях 6.1-6.14.

Таблиця 6.1 – Тестування вибору процесора

Призначення	Вибір процесора
Очікуваний результат	– додавання процесору до тимчасового комп’ютера; – виклик фільтрації доступних компонентів;
Виконання	Успішно

Таблиця 6.2 – Тестування вибору відеокарти

Призначення	Вибір відеокарти
Очікуваний результат	– додавання відеокарти до тимчасового комп’ютера; – виклик фільтрації доступних компонентів.
Виконання	Успішно

Таблиця 6.3 – Тестування вибору материнської плати

Призначення	Вибір материнської плати
Очікуваний результат	– материнська плата додана до комп’ютера; – виклик фільтрації доступних компонентів.
Виконання	Успішно

Таблиця 6.4 – Тестування вибору постійної пам’яті

Призначення	Вибір постійної пам’яті
Очікуваний результат	– постійна пам’ять додана до комп’ютера; – виклик фільтрації доступних компонентів.

Виконання	Успішно

Таблиця 6.5 – Тестування вибору оперативної пам’яті

Призначення	Вибір оперативної пам’яті
Очікуваний результат	– оперативна пам’ять додана до комп’ютера; – виклик фільтрації доступних компонентів.
Виконання	Успішно

Таблиця 6.6 – Тестування вибору блоку живлення

Призначення	Вибір блоку живлення
Очікуваний результат	– блок живлення доданий до комп’ютера; – виклик фільтрації доступних компонентів.
Виконання	Успішно

Таблиця 6.7 – Тестування додавання ПК у збережені

Призначення	Додавання готового ПК у збережені
Очікуваний результат	– готовий ПК додано у збережені, якщо користувач авторизований – відкрито вікно авторизації, якщо користувач не авторизований
Виконання	Успішно

Таблиця 6.8 – Тестування видалення збереженого ПК

Призначення	Видалення збереженого ПК
Очікуваний результат	Збережений ПК видалений
Виконання	Успішно

Таблиця 6.9 – Тестування редагування збереженого ПК

Призначення	Редагування збереженого ПК
Очікуваний результат	Збережений ПК обраний до змін
Виконання	Успішно

Таблиця 6.10 – Тестування авторизації

Призначення	Авторизація існуючого користувача
Очікуваний результат	– користувач авторизований – отримані списки збережених комп’ютерів
Виконання	Успішно

Таблиця 6.11 – Тестування реєстрації

Призначення	Реєстрація нового користувача
Очікуваний результат	– новий користувач зареєстрований – користувач авторизований
Виконання	Успішно

Таблиця 6.12 – Тестування відображення доданих користувачами компонентів

Призначення	Встановлення флагу відображення доданих користувачем компонентів
Очікуваний результат	Додавання до списків комплектуючих тих, що були додані користувачами
Виконання	Успішно

Таблиця 6.13 – Тестування видалення доданих користувачами компонентів адміністратором

Призначення	Видалення компонентів, що їх додано користувачами
Очікуваний результат	Компонент видалено зі списків
Виконання	Успішно

Таблиця 6.14 – Тестування авторизації адміністратора

Призначення	Авторизація адміністратора
Очікуваний результат	– адміністратор авторизований – відкрито доступ до відповідного меню
Виконання	Успішно

6.3 Ручне тестування

Метод ручного тестування полягає в тому, що розробник або тестувальник самостійно та вручну повторює дії, що очікується їх буде здійснювати користувач програмного забезпечення. Найчастіше, працездатність програми таким чином відстежується за змінами у інтерфейсі програми. У загальній

практиці застосовується таке поняття як бета-тестування, що реалізується відкриттям раннього доступу до продукту та збором статистичних даних щодо виникаючих помилок завдяки масовому використанню програми майбутньою аудиторією[14].

Найоптимальнішим є використання даного типу тестування на ранніх стадіях розробки, для налагодження невеликих частин коду задля переконання у їх працездатності. Під час написання, після реалізації логічного блоку коду, програма запускається та нетривалою роботою перевіряється її правильність. Варто зазначити, що ручним тестуванням неможливо задіяти усі частини коду, але можна створити таку комбінацію дій, що передбачає нелогічне (з точки зору розробника) за частотою виклику або своїм порядком задіяння різних частин програми та може викликати певні баги. Також, воно дозволяє переконатися у зручності розміщення елементів керування, правильності їх відображення та швидкодії застосунку.

7 ВПРОВАДЖЕННЯ ТА ВИКОРИСТАННЯ РОЗРОБЛЕНОГО ДОДАТКУ

7.1 Програмні та апаратні вимоги

Даний додаток було розроблено мовою C#. Мінімальні та рекомендовані апаратні вимоги базуються на обраних платформах програмування та представлені у таблиці 7.1 та 7.2.

Таблиця 7.1 – Перелік мінімальних апаратних вимог до розробленого додатку

Процесор	2-ядра, тактова частота 1.4 ГГц, кеш третього рівня 2 Mb
Місце на диску	20 Mb
Оперативна пам’ять	300 Mb
Операційна система	Windows 10, macOS, Linux

Таблиця 7.2 – Перелік рекомендованих апаратних вимог до розробленого додатку

Процесор	4-ядра, тактова частота 2.8 ГГц, кеш третього рівня 4 Mb
Місце на диску	100 Mb
Оперативна пам’ять	600 Mb
Операційна система	Windows 10, macOS, Linux

Фреймворк .NET Core потребує встановленого на користувацькому ПК спеціального програмного забезпечення, а саме:

- середа виконання .NET Core;
- Microsoft Visual C++ 2015.

7.2 Інструкція з експлуатації

Додаток запускається через файл «BuildYourPC.exe» та відкриває одразу головне вікно, з якого користувач має змогу одразу перейти до збору ПК. Рекомендовано зареєструватися, або авторизуватися у системі задля розширення функціональності за рахунок можливості зберігання ПК та додавання до системи компонентів які у ній відсутні. З діаграмою використання можна ознайомитись на кресленику ІА61.310БАК.005 Д1.

Реалізований функціонал поділяється на користувацький та адміністраторський. Користувач може:

- авторизуватися та реєструватися у системі;
- збирати ПК, перевіряти його збалансованість;
- додавати відсутні у системі компоненти;
- зберігати готові ПК у своєму акаунті;
- видаляти та редагувати збережені ПК.

Функціонал адміністратора дублює функції звичайного користувача та надає право на:

- додавання компонентів без позначки «додано користувачем»;
- видаляти компоненти, що не відповідають зазначеним у них критеріям;
- переглядати зареєстровані профілі, видаляти такі, що додають до системи неадекватні компоненти

7.3 Використання додатку

На даний момент є декілька сценаріїв використання даного застосунку, а саме:

- персональне використання у якості довідника та менеджеру;
- інтеграція додатку до інтернет-магазинів в рамках співпраці (як, наприклад, Telemart);

– використання в майстернях за умови надання ними подібних послуг як менеджер, або задля отримання первинної інформація від клієнта яким він бачить свій ПК;

– використання у навчальних закладах з метою наглядного вивчення будови персонального комп’ютера та основних характеристик його компонентів.

На етапі розробки додатку було внесено компоненти, що містяться у таблиці додатку Б. Вони були відібрані за популярністю, потужністю та принципом конкуренції, тобто базовий набір складається з такого числа та різновиду компонентів, щоб вже на начальному етапі користувач мав широкий вибір комплектуючих. Додано обидва найпопулярніших виробники процесорів Intel та AMD, по одному представнику від кожного цінового сегменту, материнські плати відповідного ним сокету та чипсету, оперативна пам’ять DDR3 та DDR4 з різною кількістю планок та значенням частоти і об’єму. Також є вибір між відеокартами різної потужності з боку NVIDIA та ATI, постійною пам’яттю різного об’єму та форм-фактору, блоків живлення у діапазоні від 500 до 1000 Ватт.

ВИСНОВКИ

Враховуючи значну кількість сервісів, що полегшують пошук та вибір тих чи інших предметів побуту є доцільним розробка автоматизованої системи надання рекомендацій з підбору комплектуючих персонального комп'ютера. У даному дипломному проєкті було спроектовано таку систему шляхом фундаментального дослідження та аналізу предметної області та розроблено її з використанням сучасних рішень. Проведено огляд та порівняння існуючих рішень, на основі їх переваг та недоліків сформовані основні вимоги до даного застосунку.

При проведенні наступних етапів розробки створено програму, що задовольняє необхідним критеріям та реалізує наступний функціонал:

– проектування ПК шляхом вибору кожного комплектуючого;

– фільтрація комплектуючих задля забезпечення їх сумісності;

– аналіз спроектованого ПК на збалансованість компонентів;

– можливість збереження ПК у персональному акаунті;

– можливість підтримання актуальності списку комплектуючих, шляхом надання користувачам прав на додавання новий компонентів до загальної бази даних.

Для забезпечення розширюваності та підтримки готового рішення використано шаблон проектування MVVM, дотримано основних принципів об'єктно орієнтованого програмування, принципів SOLID, створено гнучку модель даних. Застосовано сучасні та популярні засоби розробки програмного забезпечення, використано перелік шаблонів проектування, що спрощують реалізацію певної бізнес-логіки та зменшують зв'язність коду, а отже й полегшують підтримку. Розроблене рішення є кросплатформеним, використовує технології, що підтримуються та розвиваються корпорацією Microsoft, отже не втратять актуальності найближчим часом.

Під час розробки було впроваджено неперервне тестування, що гарантує відмовостійкість розробленого продукту та працездатність усього описаного

функціоналу. Реалізовано автоматичні unit-тести, проведено цикл ручного тестування.

Хоча дане рішення не відповідає усім вимогам, що можуть висуватися до даного програмного забезпечення, а також не інтегроване до торгівельної платформи задля максимального збереження актуальності переліку доступних компонентів воно є найоптимальнішим та найбільш дружнім до користувачів-початківців, що не є достатньо обізнаними у даній сфері та не можуть впевнено користуватися великою кількістю фільтрів. Даний додаток вирішує цю проблему шляхом автоматичного видалення з доступних до вибору компонентів тих, що не відповідають накладеним обмеженням. Таким чином, мету даного дипломного проєкту вважаю повністю досягненою, а всі поставлені завдання виконаними.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Скотт Мюллер. Модернизация и ремонт ПК = Upgrading and Repairing PCs. — 17-е изд. — М.: Вильямс, 2007. — С. 241—443. — ISBN 0-7897-3404-4.
2. Розробка додатків (WPF) [Електронний ресурс] : Режим доступу : https://docs.microsoft.com/ru-ru/dotnet/framework/wpf/app-development/ – 26.01.2018 р.
3. Документація C# [Електронний ресурс] : Режим доступу : https://docs.microsoft.com/ru-ru/dotnet/csharp/ – 26.02.2020 р.
4. Основи Entity Framework [Електронний ресурс]: Режим доступу: https://metanit.com/sharp/entityframework/3.13.php
5. Грэди Буч. Объектно-ориентированный анализ и проектирование с примерами приложений на C++ = Object-Oriented Analysis and Design with Applications / Пер. И.Романовский, Ф.Андреев. — 2-е изд. — М., СПб.: «Бином», «Невский диалект», 1998. — 560 с. — 6000 экз. — ISBN 5-7989-0067-3.
6. Edsger W. Dijkstra Программирование как вид человеческой деятельности. 1979 (EWD117)
7. Entity Framework Code First [Електронний ресурс] : Режим доступу : https://docs.microsoft.com/ru-ru/aspnet/mvc/overview/getting-started/getting-started-with-ef-using-mvc/migrations-and-deployment-with-the-entity-framework-in-an-asp-net-mvc-application – 16.01.2019
8. MVVM Frameworks [Електронний ресурс] : Режим доступу : https://docs.microsoft.com/en-us/archive/blogs/llobo/mvvm-frameworks – 30.10.2009 р.
9. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. П75 Приемы объектно-ориентированного проектирования. Паттерны проектирования. — СПб: Питер, 2001. — 368 с.: ил. (Серия «Библиотека программиста») ISBN 5-272-00355-1
10. Фаулер, Мартин. Чистый код: создание, анализ и рефакторинг. — Спб: «Питер», 2013. — 464 с. — ISBN 978-5-496-00487-9.

11. Мартин Фаулер. Шаблоны корпоративных приложений (Signature Series) – Patterns of Enterprise Application Architecture (Addison-Wesley Signature Series). — М.: «Вильямс», 2012. — 544 с. — ISBN 978-5-8459-1611-2.

12. Test-Driven Development [Електронний ресурс]: Режим доступу: https://codedream.me/2016/03/01/test-driven-development/ – 01.03.2016

13. Лайза Криспин, Джанет Грегори. Гибкое тестирование: практическое руководство для тестировщиков ПО и гибких команд = Agile Testing: A Practical Guide for Testers and Agile Teams. — М. : «Вильямс», 2010. — 464 с. — (Addison-Wesley Signature Series) — 1000 прим. — ISBN 978-5-8459-1625-9.

14. Kaner, Cem; Falk, Jack; Nguyen, Hung Quoc (1999). Testing Computer Software, 2nd Ed. New York, et al: John Wiley and Sons, Inc. с. 480. ISBN 0-471-35846-0.

15. Оформлення текстових документів у навчальному процесі. Стандарт організації (кафедри) СОУ АУТС 01-15. Для студентів кафедри автоматики та управління в технічних системах [Текст] / Уклад.: Я.Ю. Дорогий, Н.Б. Репнікова, О.І. Ролік, Л.Ю. Юрчук – К.: НТУУ «КПІ», 2015. – 27 с.

16. ДСТУ 1.5 : 2003 «Правила побудови, викладання, оформлення та вимоги до змісту нормативних документів». «Державна система стандартизації України. Загальні вимоги до побудови, викладу, оформлення та змісту стандартів» [Текст]. – На заміну ДСТУ 1.5-93 ; Чинний від 2003-07-01. – Київ : Видавництво стандартів, 2003. – 44 с.

17. Методичні вказівки до виконання дипломних робіт освітньо-кваліфікаційного рівня «бакалавр» для студентів напряму підготовки 6.050201 «Системна інженерія» кафедри «Автоматики та управління у технічних системах» [Текст] / Уклад.: К.С. Дорошенко, А. О. Новацький, Н. Б. Репнікова, Л. Ю. Юрчук. – К.: НТУУ «КПІ», 2014. – 67 с.

ДОДАТОК А

Код програми

```
public class BuildingVM : INotifyPropertyChanged
    {
        private bool _IsShowUserAdded;
        public bool IsShowUserAdded
        {
            get => _IsShowUserAdded;
            set
            {
                _IsShowUserAdded = value;
                UpdateCollections();
                OnPropertyChanged(nameof(IsShowUserAdded));
            }
        }

        #region CPU

        private string _CPUfilter;
        public string CPUfilter
        {
            get => _CPUfilter;
            set
            {
                if (!String.IsNullOrEmpty(value))
                {
                    _CPUfilter = value;
                    CPUs = new ObservableCollection<CPU>(CPUs.Where(x =>
x.Name.Contains(_CPUfilter)));
                }
                else
```

```
        {
            CPUs = DatabaseLists.GetDatabaseLists().CPUs;
        }
        OnPropertyChanged(nameof(CPUfilter));
    }
}

private CPU _TempCPU;
public CPU TempCPU
{
    get => _TempCPU;
    set
    {
        if (value != null)
        {
            _TempCPU = value;
            CPUinfo = _TempCPU.PersonalToString();
            OnPropertyChanged(nameof(TempCPU));
        }
    }
}
private string _CPUinfo;
public string CPUinfo
{
    get => _CPUinfo;
    set
    {
        if (value != null)
        {
            _CPUinfo = value;
            OnPropertyChanged(nameof(CPUinfo));
```

```
            }
        }
    }
    #endregion
 public BuildingVM(User tempUser)
    {
        UpdateCollections();
        _addCPUCommand = new RelayCommand(AddCPUOpenWindow);
        _addMotherboardCommand = new
RelayCommand(AddMotherboardOpenWindow);
        _addGraphicCardCommand = new
RelayCommand(AddGraphicCardOpenWindow);
        _addRAMCommand = new RelayCommand(AddRAMOpenWindow);
        _addMemoryCommand = new RelayCommand(AddMemoryOpenWindow);
        _addPowerSupplyCommand = new
RelayCommand(AddPowerSupplyOpenWindow);
    }
public void AddCPUOpenWindow()
    {
        var addComponentWindow = new AddComponentWindow(new
AddCPUVM());
        addComponentWindow.ShowDialog();
    }
    public void AddMotherboardOpenWindow()
    {
        var addComponentWindow = new AddComponentWindow(new
AddCPUVM());
        addComponentWindow.ShowDialog();
    }
    public void AddGraphicCardOpenWindow()
    {
```

```
            var addComponentWindow = new AddComponentWindow(new
AddCPUVM());
            addComponentWindow.ShowDialog();
        }
        public void AddRAMOpenWindow()
        {
            var addComponentWindow = new AddComponentWindow(new
AddCPUVM());
            addComponentWindow.ShowDialog();
        }
        public void AddMemoryOpenWindow()
        {
            var addComponentWindow = new AddComponentWindow(new
AddCPUVM());
            addComponentWindow.ShowDialog();
        }
        public void AddPowerSupplyOpenWindow()
        {
            var addComponentWindow = new AddComponentWindow(new
AddCPUVM());
            addComponentWindow.ShowDialog();
        }

        #endregion

        public void MakeFiltration()
        {
            if (TempCPU != null)
            {
```

```
            RAMs = new
ObservableCollection<RAM>(DatabaseLists.GetDatabaseLists().RAMs.Where(x =>
x.Age == TempCPU.RamAge));

            if (TempRAM != null)
                if (TempRAM.Age != TempCPU.RamAge)
                    TempRAM = null;
        }
    }

    private void UpdateCollections()
    {
        if (!IsShowUserAdded)
        {
            CPUs = new
ObservableCollection<CPU>(DatabaseLists.GetDatabaseLists().CPUs.Where(x =>
x.AddedByUser == IsShowUserAdded));
            GraphicCards = new
ObservableCollection<GraphicCard>(DatabaseLists.GetDatabaseLists().GraphicCard
s.Where(x => x.AddedByUser == IsShowUserAdded));
            Memories = new
ObservableCollection<Memory>(DatabaseLists.GetDatabaseLists().Memories.Where
(x => x.AddedByUser == IsShowUserAdded));
            Motherboards = new
ObservableCollection<Motherboard>(DatabaseLists.GetDatabaseLists().Motherboar
ds.Where(x => x.AddedByUser == IsShowUserAdded));
            PowerSupplies = new
ObservableCollection<PowerSupply>(DatabaseLists.GetDatabaseLists().PowerSuppl
ies.Where(x => x.AddedByUser == IsShowUserAdded));
```

```
            RAMs = new
ObservableCollection<RAM>(DatabaseLists.GetDatabaseLists().RAMs.Where(x =>
x.AddedByUser == IsShowUserAdded));
        }
        else
        {
            CPUs = DatabaseLists.GetDatabaseLists().CPUs;
            GraphicCards = DatabaseLists.GetDatabaseLists().GraphicCards;
            Memories = DatabaseLists.GetDatabaseLists().Memories;
            Motherboards = DatabaseLists.GetDatabaseLists().Motherboards;
            PowerSupplies = DatabaseLists.GetDatabaseLists().PowerSupplies;
            RAMs = DatabaseLists.GetDatabaseLists().RAMs;
        }
    }
#region GraphicCard
    private GraphicCard _TempGraphicCard;
    public GraphicCard TempGraphicCard
    {
        get => _TempGraphicCard;
        set
        {
            if (value != null)
            {
                _TempGraphicCard = value;
                GraphicCardinfo = _TempGraphicCard.PersonalToString();
                OnPropertyChanged(nameof(TempGraphicCard));
            }
        }
    }
    private string _GraphicCardinfo;
    public string GraphicCardinfo
```

```
{
    get => _GraphicCardinfo;
    set
    {
        if (value != null)
        {
            _GraphicCardinfo = value;
            OnPropertyChanged(nameof(GraphicCardinfo));
        }
    }
}
#endregion

#region RAM
private RAM _TempRAM;
public RAM TempRAM
{
    get => _TempRAM;
    set
    {
        _TempRAM = value;
        if (_TempRAM != null)
            RAMinfo = _TempRAM.PersonalToString();
        else
            RAMinfo = null;
        OnPropertyChanged(nameof(TempRAM));
    }
}
private string _RAMinfo;
public string RAMinfo
{
```

```
        get => _RAMinfo;
        set
        {
            _RAMinfo = value;
            OnPropertyChanged(nameof(RAMinfo));
        }
    }
    #endregion

    #region Memory
    private Memory _TempMemory;
    public Memory TempMemory
    {
        get => _TempMemory;
        set
        {
            if (value != null)
            {
                _TempMemory = value;
                Memoryinfo = _TempMemory.PersonalToString();
                OnPropertyChanged(nameof(TempMemory));
            }
        }
    }
    private string _Memoryinfo;
    public string Memoryinfo
    {
        get => _Memoryinfo;
        set
        {
            if (value != null)
```

```
        {
            _Memoryinfo = value;
            OnPropertyChanged(nameof(Memoryinfo));
        }
    }
}
#endregion

#region PowerSupply
private PowerSupply _TempPowerSupply;
public PowerSupply TempPowerSupply
{
    get => _TempPowerSupply;
    set
    {
        if (value != null)
        {
            _TempPowerSupply = value;
            PowerSupplyinfo = _TempPowerSupply.PersonalToString();
            OnPropertyChanged(nameof(TempPowerSupply));
        }
    }
}
private string _PowerSupplyinfo;
public string PowerSupplyinfo
{
    get => _PowerSupplyinfo;
    set
    {
        if (value != null)
        {
```

```
            _PowerSupplyinfo = value;
            OnPropertyChanged(nameof(PowerSupplyinfo));
        }
    }
}
#endregion

#region Motherboard
private Motherboard _TempMotherboard;
public Motherboard TempMotherboard
{
    get => _TempMotherboard;
    set
    {
        if (value != null)
        {
            _TempMotherboard = value;
            Motherboardinfo = _TempMotherboard.PersonalToString();
            OnPropertyChanged(nameof(TempMotherboard));
        }
    }
}
private string _Motherboardinfo;
public string Motherboardinfo
{
    get => _Motherboardinfo;
    set
    {
        if (value != null)
        {
            _Motherboardinfo = value;
```

```
                OnPropertyChanged(nameof(Motherboardinfo));
            }
        }
    }
    #endregion
```

ДОДАТОК Б

Таблиця доданих комплектуючих

Процесори	AMD Ryzen 3 3200G with Vega 8 AMD Ryzen 3 3100 AMD Ryzen 3 3300X AMD Ryzen 5 3400G with Vega 11 AMD Ryzen 5 3600 AMD Ryzen 7 3700X AMD Ryzen 9 3900X Intel Core i3-9100F Intel Core i5-9400F Intel Core i7-9700K Intel Core i9-9900K
Материнські плати	Asus Prime A320M-K Asus Prime B550M-A Asus Prime X570-Pro ASRock Z370 Pro4 Gigabyte Q370M D3H GSM PLUS Asus Prime H370-Plus Asus Prime B360-Plus Asus Prime H310M-R R2.0
Відеокарти	Asus PCI-Ex GeForce GT 1030 Phoenix OC 2GB GDDR5 Asus PCI-Ex GeForce GTX 1050 Ti ROG Strix 4GB GDDR5 Msi Pci-Ex Geforce Gtx 1080 Ti Aero Oc 11Gb Gddr5X Gigabyte PCI-Ex GeForce GTX 1650 Gaming OC 4GB GDDR5

	MSI PCI-Ex GeForce GTX 1660 Ti Gaming X 6G 6GB GDDR6
Постійна пам’ять	WD Black 1TB 7200rpm (HDD) WD Red 2TB 5400rpm (HDD) WD Blue 2TB 5400rpm (HDD) WD Purple 1TB 5400rpm (HDD) Samsung 860 Evo-Series 500GB 2.5" (SSD) Kingston SSD HyperX Fury 3D 480GB 2.5" (SSD)
Оперативна пам’ять	HyperX DDR4-3200 16384MB Kingston DDR4-2933 8192MB Kingston DDR4-2666 4096MB
Блок живлення	be quiet! Dark Power Pro 11 1000W DeepCool 500W DeepCool 700W Asus ROG Thor 850W be quiet! Pure Power 11 700W Chieftec Proton BDF-1000C 1000W